Globus Quick Start Guide

Globus Software Version 1.1
November 1999
Preliminary Draft

the globus project
www.globus.org

Globus Quick Start Guide

Globus Software Version 1.1
November 1999
Preliminary Draft

the globus project
www.globus.org

The Globus Project is a community effort, led by Argonne National Laboratory and the University of Southern California's Information Sciences Institute. Globus is developing the basic software infrastructure for computations that integrate geographically distributed computational and information resources.

This document was produced as a community effort by the Information Power Grid (IPG) group at NASA Ames Research Center, Code IN; Argonne National Laboratory; the National Computational Science Alliance; and the University of Southern California's Information Sciences Institute.

This Globus Quick Start Guide is available online at
www.globus.org/documentation/quick_start.html

Contents

Chapter 1 Extremely Quick Start....1
Chapter 2 Introduction....2
Summary of Chapter....2
What are Grids?....2
What is Globus?....2
Globus Features....3
Testbeds....3
Four Levels of the Grid....4
The Integrated Grid Architecture....4
Grid Fabric: Layer One....4
Grid Services: Layer Two....4
Application Toolkits: Layer Three....5
Specific Applications: Layer Four....5
Purpose of This Document....5
Support and Usage Help....5
About this Guide....6
Audience....6
Assumptions....6
Organization and Conventions....6
Chapter 3 Globus Certificate....7
Summary of Chapter....7
Why You Want a Certificate....7
Local Accounts....7
Set Your Environment and Path....8
Globus Certification: Using grid-cert-request....8
Easy Steps to Certification....9
Checking Version....10
Ready to run....10
Certificate Test....10
Chapter 4 Proxy Credential....11
Summary of Chapter....11
Obtaining a Proxy Credential....11
Using the grid-proxy-init command....11
Chapter 5 Running a Job....12
Summary of Chapter....12

Commands to Run a Job....12
globus-job-run....12
globus-job-submit....12
globusrun....12
mpirun....12
Test: Ready to Run?....12
globus-job-run: Hello World....13
globus-job-run Examples....13
globus-job-submit....14
globus-job-submit Example....14
Other globus-job-* Commands....14
Globus Resource Specification Language (RSL)....15
Writing a Simple RSL: Hello Globus World....15
RSL Syntax....16
Letting globus-job-run write the RSL for you....16
Using the globusrun Command....16
More globusrun Examples....16
Checking and Killing Jobs....17
What is My Job id?....17
Is My Job Running?....17
Killing a Job....17
What Happens When You Submit a Job....17
Global Resource Allocation Manager (GRAM)....18
The Dynamically Updated Request Online Co-allocator (DUROC)....18
Chapter 6 GIS, Grid Information Service (MDS)....19
Summary of Chapter....19
Terminology....19
Tools....19
grid-info-search....19
Java-Browser....20
CGI Browser....20
LDAP in the Community....20
Chapter 7 GASS: Remote Data....21
Summary of Chapter....21
globus-gass-server....21
globus-url-copy....21
globus-rcp....21
Chapter 8 RSL With Shell Script....22
Summary of Chapter....22

The RSL File....22
The Shell Script....22
Chapter 9 Parallel Examples....23
Summary of Chapter....23
Preliminary Steps (For MPI examples)....23
Running an MPICH-G Program on the NAS IPG Testbed....23
A Basic MPI Example Using MPICH-G....23
An MPI Example on Two Hosts with Global Synchronization....24
Specifying additional attributes using an RSL....25
Chapter 10 Globus Commands....27
Summary of Chapter....27
Globus 1.1 Tools....27
Security....27
Job Submission....27
Information Services....28
Other Tools....28
Chapter 11 Bibliography and Reference....30
Web and Email Resources....30
Technical Papers....30
The Grid Book....30
Chapter 12 Glossary....31

Chapter 1 Extremely Quick Start

Here are the briefest instructions for getting started with Globus version 1.1.

1. Get accounts on globus-enabled machines; e.g., email accounts@nas.nasa.gov
2. Login to a globus-enabled machine.
3. Setup your environment and path, if necessary; see page 8.
4. Type grid-cert-request. Follow instructions. Be sure to send your subject name from the grid-cert-request response to the system administrators of all your accounts on which you plan to use Globus.
5. Wait 2 days or less to receive your signed certificate via email. Follow the instructions in the email for how to save your certificate.
6. Type globus-setup-test. If that gives you "Success!" then:
7. Type grid-proxy-init.
8. Type globus-job-run <*hostname*> /bin/echo "Hello World."
9. Type grid-proxy-destroy
10. Use <*globus command*> -help for help

Chapter 2 Introduction

Summary of Chapter

Grids are super internets for high performance computing: world-wide collections of high end resources—such as supercomputers, storage, advanced instruments, and immersive environments. These resources and their users are often separated by great distances and connected by high speed networks.

The Globus development team has created a set of underlying Grid services and a software toolkit for using the geographically distributed resources on Grids.

This chapter contains background information only, no instructions. This manual assumes you know Unix. The "%" sign represents the Unix prompt in examples.

What are Grids?

Grids bring together geographically and organizationally dispersed computational resources, such as CPUs, storage systems, communication systems, real-time data sources and instruments, and human collaborators. If you are developing an application that requires geographically distributed high-end resources, you will want to know about Grids. Grids are new—many of the enabling technologies have not yet been invented. If you are reading this in the early 2000s, you are one of the Grid pioneers.

Current testbed grids will eventually become "the Grid." In much the same way that the nationwide electric power grid connects sources of electricity, the information Grid will connect sources and users of high-performance computing cycles, allowing these cycles to be generated in one place and used in another. Remote collaborators will work together on large-scale projects. Scientists will be able to access extremely data-intensive instruments from across the country, in real time.

The goal of the Grid community is to provide dependable, consistent, pervasive access to (high end) resources.

What is Globus?

The Globus software toolkit facilitates the creation of usable Grids, enabling high-speed coupling of people, computers, databases and instruments. With Globus, you can run your gigabyte-per-time-step dataset job on two or more high performance machines at the same time, even though the machines might be located far apart and owned by different organizations. Globus software helps scientists deal with very large datasets and complex remote collaborations.

Globus software is used for large distributed computational jobs, remote instrumentation, remote data transfer, and shared immersive spaces. For example,

- Globus used in live runs at world's brightest x-ray source : http://www.mcs.anl.gov/~laszewsk/xray/cmt/gallery/
- Distributed supercomputing, smart instruments, Teleimmersive Applications: http://www.globus.org/overview/applications.html

Globus Features

Globus is designed to offer features such as uniform access to distributed resources with diverse scheduling mechanisms; information service for resource publication, discovery, and selection; API and command-line tools for remote file management, staging of executables and data; and enhanced performance through multiple communication protocols.

Testbeds

Grids are a community effort. University, commercial, and government computer science communities contribute to the development of Grids. Several significant testbeds are up and running. You can obtain accounts by sending email to the addresses given below. Include your who-why-where: full name, username if you have one on that system, company or affiliation, and brief description of work.

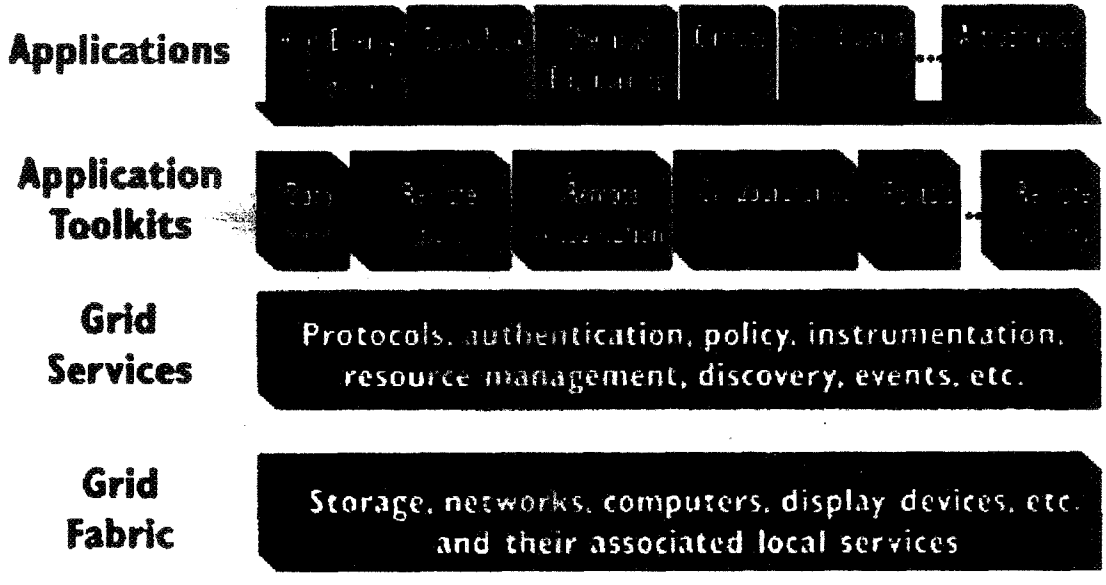
Bear in mind that these Grids are works-in-progress; things still break.

Each of these Grids uses Globus services to provide uniform access to a large collection of resources for a particular Grid community. Work is ongoing to enable interoperation of these various testbeds; however, most likely you will be using one particular testbed. There are other testbeds in progress.

- GUSTO—this is the original testbed for the Globus software. All resources that have Globus software installed are part of the GUSTO Grid. Resources include high-performance computers , instruments, and immersive environments. See http://www.globus.org/testbeds To apply for accounts, contact individual sites.
- IPG, the Information Power Grid— this is NASA's testbed for Grid research. See www.nas.nasa.gov/IPG. To apply for accounts, send email to accounts@nas.nasa.gov
- National Technology Grid—this is the testbed created by the National Computational Science Alliance, led by NCSA. See http://www.ncsa.uiuc.edu/alliance/g-force/ To apply for accounts, send email to allocations@ncsa.uiuc.edu
- The National Partnership for Advanced Computational Infrastructure (NPACI), led by the San Diego Supercomputing Center, operates a testbed. See http://www.npaci.edu/

Four Levels of the Grid

The Integrated Grid Architecture



Grid Fabric: Layer One

The *fabric* of the Grid comprises the underlying systems, computers, operating systems, networks, storage systems, and routers—the underlying building blocks.

Grid Services: Layer Two

Grid services *integrate* the components of the Grid fabric. Examples of the services that are provided by Globus:

GRAM

The Globus Resource Allocation Manager, GRAM, is a basic library service that provides capabilities to do remote-submission job start up. GRAM unites Grid machines, providing a common user interface so that you can submit a job to multiple machines on the Grid fabric. GRAM is a general, ubiquitous service, with specific application toolkit commands built on top of it.

GIS

The Grid Information Service, GIS, formerly known as the Metacomputing Directory Service, MDS, provides information service. You query GIS to discover the properties of the machines, computers and networks that you want to use: how many processors are available at this moment? What bandwidth is provided? Is the storage on tape or disk? Is the visualization device immersive desk or cave? Making use of an LDAP server, GIS provides middleware information in a common interface, putting a unifying picture on top of disparate equipment. See page 19.

GSI

The Grid Security Infrastructure, GSI, is a library for providing generic security services for applications that will be run on the Grid. Application programmers use the gss-api library for adding authentication to a program. GSI provides programs, such as grid-proxy-init, to facilitate login to a variety of sites, while each site has its own flavor of security measures . That is, on the fabric layer, the various machines you want to use might be governed by disparate security policies; GSI provides a means of simplifying multiple remote logins.

Application Toolkits: Layer Three

Application toolkits use Grid Services to provide higher-level capabilities, often targeted to specific classes of application.

For example, the Globus development team has created a set of Grid service tools and a toolkit of programs for running remotely distributed jobs. These include remote job submission commands (globusrun, globus-job-submit, globus-job-run), built on top of the GRAM service and MPICH-G, a Grid-enabled implementation of the Message Passing Interface (MPI).

A number of groups are also developing a range of other toolkits, for example to support the distributed management of large datasets, collaborative visualization, and online instrumentation. These are not discussed in this document.

Specific Applications: Layer Four

A rich variety of applications can and have been developed that build on services provided by the three layers just described. For example, OVERFLOW is a thin-layer Navier-Stokes flow solver for structured grids that has been modified to operate on multiple supercomputers via the use of MPICH-G; see page 22 for an example of running OVERFLOW through Globus.

Other application areas being targeted to the Grid include high energy physics, cosmology, chemical engineering, climate, combustion and astrobiology.

Purpose of This Document

This Quick Start User Guide tells the reader how to get started using Globus Grid services and the tools built on top of those services to build applications to exploit geographically distributed resources. The instructions herein are brief and do not take you much past "Hello World," but do provide the essentials required to get started. See the Globus User Manual (under development) for more details.

Support and Usage Help

Address questions to your local system administrator. See also: Bibliography and Reference, near the end of this document.

For usage information about any Globus command, type the command with the -usage or -help options:

```
% <globus-command-name> -usage
% <globus-command-name> -help
```

Also, all programs have a man page with more elaborate exxplanations than provied by -usage. See http://www.globus.org/v1.1/programs/<*globus-command-name*>.html

About this Guide

Audience

This guide is intended for novice Globus users.

Assumptions

Before you start using Globus, you will need to fulfill these basic requirements.

You Know Basic Unix

This manual assumes that you know at least the fundamentals of Unix. You can find good Unix information at http://www.ugu.com/sui/ugu/show?help.beginners

Globus Is Installed

If you are planning to use the resources on the existing Grid testbeds, you probably will find Globus already installed. If you want to add your resources to a testbed, ask your system administrator to see http://www.globus.org/software/installation.html

Globus provides distinct server and client software . Server software is intended for machines to which remote access is desired; in current testbeds, server software is often installed only on high-end resources. Client software allows users to access remote servers; it is installed on workstations, laptops, and other desktop machines. If you want to run Globus from your desktop machine, you can login to a Globus machine, or install Globus *client* software on your machine.

Globus commands start with *globus* or *grid*.

Organization and Conventions

The chapters are presented in step-by-step sequence. Only the bare essentials are included. A glossary at the end defines terms.

Conventions used in this guide:

% Unix prompt; do not type

$ Proceeds a variable

<*italics*> Substitute your own information. For example, <*src-dir*> means enter in your source directory pathname. Italics are also used within plain text for emphasis.

[ ] Square brackets indicate parameters that sometimes may be omitted.

Resources Refers to computers, instruments, or other machines (as opposed to data)

Chapter 3 Globus Certificate

Summary of Chapter

In this chapter, you obtain a Globus Certificate in order to have authenticate-once access to all of the machines you want to use on the Globus testbed. Your Globus Certificate is like a passport; it establishes your identity on the Grid. This is what you do:

- Obtain accounts on geographically distributed machines, and path names to Globus tools.
- Check, and if necessary, set your GLOBUS_INSTALL_PATH environment variable and your path to Globus tools.
- Initiate the grid-cert-request command.
- Email all system administrators (for those resources that you wish to access via Globus tools) with your subject name—your unique Grid identifier.
- Save your usercetrt.pem and change the permissions as instructed in the grid-cert-request response.

Why You Want a Certificate

With a Globus certificate, you only have to enter your password once per Globus session. The Globus certificate is like a passport; you take the time to establish your identity to the proper authorities, and then you are issued a certificate and private key; these give you geographically distributed access. You do not have to reestablish your identity at each "border."

Local Accounts

Before using Globus, you will need to get accounts on all of the Grid computers you plan to use. Talk to your local system administrators, if your local system is part of the Grid. See Testbeds, in the previous chapter, for some places to start.

You should have one computer, running Globus client software, that you consider your home/local machine. Get your certificate from your local home machine. That is, the Certificate Authority (CA) will consider your home machine to be the machine from which you apply for your certificate.

Set Your Environment and Path

This is a critical step! Perhaps your kind and thoughtful system administrators have done it for you. To find out, type:

```
% which globus-job-submit
```

If you get a response that looks something like:

```
/software/globus-1.1.0/release/tools/mips-sgi-irix6.5/bin/globus-job-submit
```

your path and environment variable probably have been set for you. If not, you will have to work with your system administrators to get it right. You need to set the GLOBUS_INSTALL_PATH environment variable, and add the globus tools directory to your default search path. This path varies per system.

Caution: your previous Globus settings may create havoc. Get rid of them.

You can use the following commands within your Unix shell, or add them to your own dot files. Substitute *<path>* with the path to where Globus is installed on your system:

(csh, tcsh):

```
setenv GLOBUS_INSTALL_PATH <path>
source $GLOBUS_INSTALL_PATH/etc/globus-user-setup.csh
```

(sh, bash, ksh):

```
export GLOBUS_INSTALL_PATH=<path>
. $GLOBUS_INSTALL_PATH/etc/globus-user-setup.sh
```

Look at your path by using the echo command. Check to see that there is no older Globus path preceding the new one; if there is an older path, remove it or move it to after the new one. Failure to fix this would give you older versions of Globus commands. Also check to see that there is a dot in your path (preferably at the end), to indicate current directory.

```
% echo $PATH
% echo $GLOBUS_INSTALL_PATH
```

Globus Certification: Using grid-cert-request

Globus provides authenticate-once capability: a common certificate/key security method that allows single sign-on anywhere a user has accounts on the grid. In other words, after you have obtained a Globus certificate, one login will give you access to all the Grid resources on which you have accounts. This is an important feature, because it eliminates the nuisance of reentering your login and password every time you utilize another resource.

This section contains instructions for obtaining a certificate from the Globus certificate authority (CA). However, some organizations have created their own certificate authority, and may follow different procedures for requesting a certificate. Please check with your systems administrator to determine if alternate instructions should be followed when requesting a certificate.

Easy Steps to Certification

Step 1—Login

Login to your home computer in the Grid. See "Local Accounts" page 7 for clarification.

Step 2—grid-cert-request

Execute the grid-cert-request command.

```
% grid-cert-request
```

Step 3—passphrase

Follow all instructions presented by the grid-cert-request program. When asked to enter a passphrase, use at least 8 characters, including mixed-case alpha characters and at least one numeral. CAUTION: you are the only person or thing that knows your passphrase. Do not forget it! No one can retrieve it for you later. If you forget your passphrase, you will have to get a whole new certificate.

Step 4—email

An important part of the grid-cert-request command output is as follows:

```
Please e-mail the certificate request to the Globus CA
cat /u/your_user_name/.globus/ usercert_request.pem | mail ca@globus.org
```

That is, the grid-cert-request will create a dot-globus directory in your home directory (i.e., ~/.globus) and a file called usercert_request.pem within the dot-globus directory. grid-cert-request also supplies you with the Unix command for mailing the file to the Globus certificate authority (CA): use the line starting with "cat" and containing your own user name. After you email the request, you should get your certificate within two days. Send the mail from the machine on which you normally receive mail; if you send from a machine that cannot receive mail, you will get no response. If you get no response, send a note, not a new certificate request, to CA@globus.org

Step 5—email again (and again)

When you receive email from the Globus certification authority in response to your grid-cert-request email, read the response carefully and follow the directions therein.

Have your unique Grid identifier added to the grid-mapfile at each resource (machine) you intend to use. To do this, email your *subject name* to your Globus Grid system administrators. Your *subject name* is near the top of the grid-cert-request email response. For example:

```
Enclosed is your Globus Certificate for:
"/C=US/O=Globus/O=NASA Ames Research Center/OU=Numerical
Aerospace Simulation/CN=Lucy McGillicutty"
```

Email the line within the quotes (starting with /C=) to the system administrator for *each* machine you plan to use, that is, all the Grid machines on which you have accounts. To find email addresses of system administrators, see Current Testbed Utilization and Browse GIS Contents at http://www.globus.org/testbeds or talk to you local system administrator.

Step 6—save

Save the usercert.pem in your dot-globus directory. Here is an example of part of one grid-cert-request output:

Save this e-mail message into the following file.
/u/*mcgillicutty*/.globus/usercert.pem

Substitute your own path/username; that is, use the exact path in *your* email message. If you are asked if you want to overwrite usercert.pem, enter *yes*. This file contains your signed public key, now saved where Globus knows to find it.

Step 7—permissions

Change permissions on .pem files in your dot-globus directory. Enter ls -l to list your files. Use the chmod command to change permissions:

```
% chmod 400 userkey.pem
% chmod 444 usercert.pem
```

Checking Version

You will run into trouble with the commands in this book if you are not running Globus 1.1.

```
% globus-version
```

If you are not getting version 1.1.0, see Setting Environment and Path, page 8.

Ready to run

You are done. Your system administrator will let you know when everything is in place, that is, when your name has been added (by humans) to the grid-mapfile (s) on your Grid resources. See the next section for how to check if you are in grid-mapfile. After you have successfully completed the certificate test, below, you will need to get a grid-proxy, which is easy and instant: see the next chapter. Don't forget that passphrase!

As you run into bumps in the road, remember that you are a Grid pioneer. Do not expect all the roads to be paved. Grids do not yet run smoothly.

Certificate Test

On a Grid computer, enter:

```
% globus-setup-test
```

If you get Authentication test .....Failure!, your unique identifier has not yet been added to the grid-mapfile, or there may be a system problem. Talk to the system administrator if you have a problem.

Chapter 4 Proxy Credential

Summary of Chapter

In this chapter you obtain a Globus proxy by typing grid-proxy-init. The proxy gives you authenticate-once capability for 12 hours (default). While your proxy is active, you can log into any Grid resource without reentering your passphrase.

The command grid-proxy-info displays contents/status of your proxy certificate.

Obtaining a Proxy Credential

When you have: obtained accounts on some Grid testbed machines, and followed the grid-cert-request procedures, you are ready to get a proxy. The Globus proxy is based on a so called "public key" security method that allows single sign-on anywhere a user has accounts on a Grid. One proxy is good for 12 hours (default) . When you start another session on another day, get a new proxy.

Using the grid-proxy-init command

At your system prompt, enter grid-proxy-init and you will be prompted for your Globus passphrase. (Don't enter the %).

```
% grid-proxy-init
returns: Enter PEM pass phrase:
.          ..........+++++
.          ..........+++++
%
```

You can set options with command line arguments to grid-proxy-init; type the command with the -usage option to see the possibilities. For example, twelve hours is the default time for the proxy to be in effect; to set the time otherwise, use grid-proxy-init -hours <nhours>. While your proxy is in effect, you do not have to enter your passphrase on any Grid machine. If your proxy expires, simply rerun grid-proxy-init to create a new proxy.

grid-proxy-destroy

To get rid of your proxy when you are finished with it, enter grid-proxy-destroy. The proxy will self-destruct after 12 hours (or whatever duration you set), but it is a good idea to destroy it yourself when you are finished.

grid-proxy-info

Use grid-proxy-info to display the contents or test the status of your proxy.

Chapter 5 Running a Job

Summary of Chapter

In this chapter you will run a few simple Globus jobs, using globus-job-run and globus-job-submit. Then you will write an RSL (Resource Specification Language) script to use with globusrun. Remember that you must have a valid proxy before running Globus.

Commands to Run a Job

globus-job-run

globus-job-run is the basic command for running Globus jobs. globus-job-run runs in the foreground and defaults to sending output to your terminal. In its basic form, it is roughly equivalent to rsh, but also has considerably more functionality for running complex jobs on the Grid.

globus-job-submit

globus-job-submit is for submitting jobs to a remote *batch job scheduler* such as PBS. With globus-job-submit, you can submit a job, log out, and log back in later to collect the output. That is, globus-job-submit runs in the background and defaults to sending output to the machine running the command.

globusrun

globusrun is for submitting jobs that are specified using RSL, the Resource Specification Language. It can run jobs either in the foreground or background, and can send output to your terminal or to the machine running the command. The trend in Globus software development is toward considering globusrun as middleware, which can be used by application specific shell scripts to manage job submission. In fact, globus-job-run and globus-job-submit are simply shell scripts which use globusrun for job submission, but present a simpler interface to users.

mpirun

MPICH-G, a implementation of the Message Passing Interface (MPI) standard based on Globus, has an mpirun command that can be used to submit MPI parallel jobs to a multiple computers in a Grid. See page 23.

Test: Ready to Run?

Before running a Globus job, try this command to see if you have a current (today's) proxy.

```
% grid-proxy-info -all
```

To see also if your resource is available and you are in the grid-mapfile:

```
% globusrun -a -r <your_resource>/jobmanager-fork
```

For example:

```
% globusrun -a -r evelyn.nas.nasa.gov/jobmanager-fork
```

If you get, "ERROR: resolving resource manager," GIS (MDS) may be malfunctioning.

globus-job-run: Hello World

The basic syntax of globus-job-run is the same as rsh:

```
% globus-job-run evelyn.nas.nasa.gov /bin/echo "Hello World."
returns Hello World
```

This example assumes you have set your path and environment variable (page 9) so that you can start your command line with a Globus command, and that you have created a proxy using grid-proxy-init. First comes the Globus command (globus-job-run), then the hostname of the machine to run the command on (for example, evelyn.nas.nasa.gov), then the program to run (the Unix echo program) and then the argument (Hello World) for the program to use. Hello World is the output of the program. When the job is complete globus-job-run terminates.

See also: http://www.globus.org/v1.1/programs/globus-job-run.html for examples.

globus-job-run Examples

These examples show how to use globus-job-run with a set of simple hello-world programs that are *not* part of the Globus installation. Substitute your own files and filenames.

```
% globus-job-run denali.mcs.anl.gov /tmp/hw 2 5
returns: Hello from denali.mcs.anl.gov, sum = 7
```

Start the executable /tmp/hw on denali with two arguments that the executable sums up.

```
% ls
returns: hw.c hw.o hw.irix*
% globus-job-run denali.mcs.anl.gov -stage hw.irix 4 6
returns: Hello from denali.mcs.anl.gov, sum = 10
```

Stage the executable hw.irix, which resides on the local workstation, over to the remote machine, execute it and automatically remove the staged copy after the program has finished.

```
% globus-job-run -args 3 5 \
-: -np 2 evelyn.nas.nasa.gov mpi-hw.irix \
-: -np 2 pitcairn.mcs.anl.gov mpi-hw.solaris \
-: denali.mcs.anl.gov mpi-hw.irix 1 2
```

returns:

```
my_id 0 numprocs 5  I am evelyn.nas.nasa.gov, sum = 8
my_id 1 numprocs 5  I am evelyn.nas.nasa.gov, sum = 8
my_id 2 numprocs 5  I am pitcairn.mcs.anl.gov, sum = 8
my_id 3 numprocs 5  I am pitcairn.mcs.anl.gov, sum = 8
my_id 4 numprocs 5  I am denali.mcs.anl.gov, sum = 3
```

Start a multi-request with 3 subjobs. The -: delimiter denotes the start of each subjob. If no path is given, the executables are assumed to reside in the user's home directory on each of the remote hosts. (The executable in this example is compiled with MPICH-G.)

Note that some parameters can be given job-wide (such as the -args option). In addition, these parameters can be overridden at a per-subjob level. In the above example, the executable running on denali gets two other arguments to sum up.

globus-job-submit

globus-job-submit is similar to globus-job-run, but is intended for batch job submission. That is, you can remotely submit a job to some local scheduling manager like PBS, log out, and log back in later to collect the output. globus-job-submit defaults to caching the output on the remote machine, for later manual retrieval using the globus-job-get-output command. Via a command line interface, you can submit jobs either to a single resource or to a collection of resources. To see the options use the command globus-job-submit –usage.

See also: http://www.globus.org/v1.1/programs/globus-job-submit.html

globus-job-submit Example

```
% globus-job-submit ico16.mcs.anl.gov -np 32 -maxtime 120 hw.aix 6 8
returns: https://ico16.mcs.anl.gov:60106/17916/942265377/
%
```

Submit a request, for 120 minutes of compute time on 32 nodes, to the scheduler on the IBM SP at ANL. The https url returned by globus-job-submit is the job contact string, which uniquely identifies the job. The extra % denotes that this command returns immediately after submission.

globus-job-submit will cache the output at the remote site (you can override this by specifying -stdout and -stderr).

Other globus-job-* Commands

```
% globus-job-status https://ico16.mcs.anl.gov:60106/17916/942265377/
returns: ACTIVE
```

Query the status of the job. The different states are PENDING (waiting in the queue), ACTIVE (running), SUSPENDED, DONE and FAILED.

```
% globus-job-get-output https://ico16.mcs.anl.gov:60106/17916/942265377/
returns:
my_id 0 numprocs 32, sum = 14 : now sleeping for 90 minutes
my_id 1 numprocs 32, sum = 14 : now sleeping for 90 minutes
my_id 2 numprocs 32, sum = 14 : now sleeping for 90 minutes
[...]
```

Retrieve the cached output from the job. You can do this while the job is running.

```
% globus-job-cancel https://ico16.mcs.anl.gov:60106/17916/942265377/
```

returns:

```
Are you sure you want to cancel the job now (Y/N) ? y
Job canceled.
NOTE: You still need to clean files associated with the job by running globus-
job-clean <jobID>
```

Cancel a running job. Note that cached output from the job is not removed: you can still use globus-job-get-output to retrieve it after you have cancelled the job.

```
% globus-job-clean https://ico16.mcs.anl.gov:60106/17916/942265377/
```

returns:

```
WARNING: Cleaning a job means:
- Kill the job if it still running
- Remove the cached output on the remote host
Are you sure you want to cleanup the job now (Y/N) ? y
Cleanup successful.
```

Cancel the job if it still running, and remove the cached output from the remote host.

Globus Resource Specification Language (RSL)

RSL is the underlying language that Globus uses to specify resources needed for a Globus job. The globus-job-run and globus-job-submit commands hide this details of this language from the normal user. However, advanced users may want to use RSL directly handle complicated resource descriptions.

The globusrun command supports similar functionality as both globus-job-run and globus-job-submit, but uses RSL to describe jobs. Normally you write a resource specification in RSL and save it as a file. Then you run globusrun, passing it the name of the RSL file as an argument, to run the job described by the RSL.

Writing a Simple RSL: Hello Globus World

The RSL allows resources to be specified as a logical expression of *<attribute, value>* pairs. Here is a sample RSL that uses the Unix "echo" program to print "Hello Globus World!" to your screen. Create a file named hello.rsl and enter the following:

```
& (count=1)
  (executable=/bin/echo)
  (arguments="Hello Globus World!")
```

&	Indicates start of RSL script. & is used as the first character for GRAM jobs (only one resource). DUROC jobs (multiple resources) start with +
(count=1)	How many processors do you plan to use? (count=4) would run 4 separate processes, and result in 4 Hellos. Try it.

(executable=/bin/echo)	This calls the normal Unix "echo" command.
(arguments="Hello Globus World!")	This supplies the argument to echo.

RSL Syntax

See http://www.globus.org/gram/rsl_spec1.html for a complete description of RSL syntax.

Letting globus-job-run write the RSL for you

You can get globus-job-run to write your RSL; just use the globus-job-run command as above, with -dumprsl as the first option.

```
% globus-job-run -dumprsl evelyn.nas.nasa.gov /bin/echo "Hello Globus World!"
```

Using the globusrun Command

To run the hello.rsl script shown above, use globusrun (don't enter the %):

```
% globusrun -s -r <host name>/<service name> -f hello.rsl
```

For example:

```
% globusrun -s -r evelyn.nas.nasa.gov/jobmanager-fork -f hello.rsl
```

Explanation: after the Unix prompt (% in the example), enter globusrun with the -s option to send output to stdout. Use the -r option to indicate that a resource name follows. The -f option indicates a filename follows, then your filename.

Hello Globus World! ought to appear on your screen. Are having problems? Did you use grid-proxy-init ? globusrun requires the existence of a valid proxy to function correctly.

A resource name takes the form <host name>/<service name>. A <service name> usually takes the form jobmanager-<scheduler type>. For example, the service name typically used when simply forking a job on a remote resource is "jobmanager-fork". A service name of "jobmanager-pbs" would typically be used to submit a job to a PBS local scheduler. The resource name is resolved by the GIS, Grid Information Service.

globusrun takes your RSL script and parcels it out to specified resources. The RSL script by itself is a resource specification, not an executable.

In addition to starting jobs, globusrun can be used to list previously started jobs, query status of previously started jobs, parse RSL request strings, and perform authentication tests to GRAM gatekeepers. In fact, the globus-job-run and globus-job-submit programs simply use globusrun to implement much of their functionality.

More globusrun Examples

Example One

```
% globusrun -s -r pitcairn.mcs.anl.gov/jobmanager-fork
'&(executable=my_prog)'
```

Resolve the contact string for the fork jobmanager service on host pitcairn, and then submit the program "my_prog" to that resource. The standard output and standard error of my_prog will be displayed by globusrun.

Example Two

```
% globusrun -s -r ico16.mcs.anl.gov/jobmanager-easymcs
'&(executable=$(GLOBUSRUN_GASS_URL)/usr/local/my_prog)
(stdout=/tmp/output1)(stderr=/tmp/error1)(count=3)
```

Submit three copies of the local program /usr/local/my_prog to the EASY scheduler at ico16.mcs.anl.gov, and send the output to the remote /tmp directory.

Example Three

```
%globusrun -l | awk '/my_prog/ {print $1}'
```

List the job IDs of jobs I've submitted containing the string my_prog

Example Four

```
%globusrun -w -r pitcairn.mcs.anl.gov/jobmanager-fork
'&(executable=$(GLOBUS_TOOLS_PATH)/bin/globus-url-copy")
(arguments="file:/tmp/myout.1" $(GLOBUSRUN_GASS_URL)/tmp/myout.1)
```

Copy a file from pitcairn to the local host, using the GASS transfer protocol. The -w option gives the remote process write permission on the local GASS server started by globusrun. Note that as colon (:) is a reserved RSL character, the first argument must be quoted See http://www.globus.org/gram/rsl_spec1.html for more information."

Checking and Killing Jobs

What is My Job id?

Use globusrun -list or -l

Is My Job Running?

Use globus-job-status, or globusrun -list. In general, if your Unix prompt hasn't returned after a globus-job-run command, your job is still running. You can log on to the remote resource you specified and use a Unix command like ps.

Killing a Job

If you interrupt the globusrun process (by typing CTRL-C or sending it a SIGINT), your jobs should automatically be canceled.

The job cancellation routines (triggered by CTRL-C or SIGINT the globusrun process) have a timeout to allow an intelligent job to cleanup. Therefore, you should expect some delay before the globusrun process exits.

What Happens When You Submit a Job

This is a complicated process, but in streamlined form it looks like this:

1. The user runs globusrun with an RSL script (or globus-job-run or globus-job-submit), and specifies where to run.

2. globusrun contacts something called a "gatekeeper" on the remote host and performs mutual authentication. This means that the remote host knows who you are, and you know who the remote host is.

3. The gatekeeper (on the remote host) contacts a job manager service with your request. The job manager will decide how to run your job depending on the service name that is used. If you specified jobmanager-fork, the job manager will simply run the job immediately using a Unix fork (hence the name). If you specified a job scheduler (e.g., jobmanager-pbs), your job will be submitted to the scheduler and will run as the scheduler allows.

4. Your job completes.

Global Resource Allocation Manager (GRAM)

The Globus Resource Allocation Manager (GRAM) authenticates and processes the requests for resources for remote application execution, and allocates the required resources. It also returns updated information regarding the capabilities and availability of computing resources to the. GIS, Grid Information Service.

GRAM provides an API for submitting and canceling a job request, as well as checking the status of a submitted job. The specifications are written by the user in the Resource Specification Language (RSL), and are processed by GRAM as part of the job request.

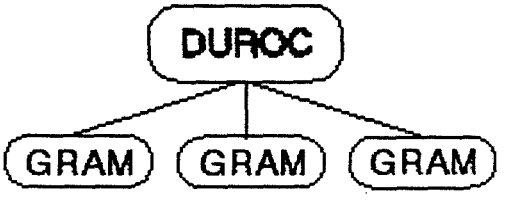
Your program can call the GRAM C API directly and supply it with an RSL, or you can use globusrun, giving it the RSL, as in the Hello World example:

```
% globusrun -s -r evelyn.nas.nasa.gov/jobmanager-fork -f  hello.rsl
```

There is a complete GRAM tutorial at
http://www.globus.org/gram/tutorial_gssapi_sslcay.html

GRAM manages jobs that use only one resource; DUROC co-allocates multiple resources. For more information, see http://www.globus.org/gram and http://www.globus.org/duroc/

The Dynamically Updated Request Online Co-allocator (DUROC)



Use DUROC to manage multiple GRAM jobs. See http://www.globus.org/duroc

Chapter 6 GIS, Grid Information Service (MDS)

Summary of Chapter

GIS is a service that allows the storage of information about the state of the Grid infrastructure. GIS is a new term for what has been known as MDS. Use the Globus grid-info commands to gather information from the GIS. GIS is work in progress; there are still problems to resolve. This chapter does not address initialization or population of information into the GIS.

Terminology

GIS is a service that allows the storage of information about the state of the Grid infrastructure. One of its services is to publish information via LDAP (lightweight directory access protocol).

The GIS information service has the ability to function as a white pages directory—for retrieving information associated with a particular name ("distinguished name"). Examples for such lookups are the retrieval of an IP number or the amount of memory associated with a particular machine. The GIS also functions as a yellow pages directory— for retrieving a list of categorized entities. Such categories are defined by "object classes." Examples of such categories are lists of computers or people.

Search constraints: a powerful extension to the white and yellow page function of the directory is the ability to augment the lookups with sophisticated Boolean search filters.

For more information, see: http://www.globus.org/mds

Tools

There are many ways to access the GIS information, including:

grid-info-search

The command grid-info-search allows searches on the GIS server based on search filters that conform to LDAP searches . The format is:

```
grid-info-search [ options ] <search filter> [attributes ]
```

To see the options available, type:

```
grid-info-search -help
```

To list all distinguished names along with all attributes of the compute resources that are stored in the directory under Argonne National Laboratory:

```
% grid-info-search -b "o=Argonne National Laboratoy, o=Globus, c=US"
"(objectclass=GlobusComputeResource)"
```

To list all distinguished names of the compute resources that are stored in the directory under Argonne National Laboratory, along with the operating system type of each:

```
% grid-info-search -b "o=Argonne National Laboratoy, o=Globus, c=US"
"(objectclass=GlobusComputeResource)" dn ostype
```

To list all distinguished names of the compute resources that are stored in the directory under Argonne National Laboratory that have the attribute ostype with a value equal to irix:

```
% grid-info-search -b "o=Argonne National Laboratoy, o=Globus, c=US"
"(&(objectclass=GlobusComputeResource)(ostype=irix))" dn
```

If one or more entries are found, that meet the search filter constraint, each entry is written to standard output and separated with a single blank line. To restrict the attributes that are output, list them at the end of the grid-info-search command. Search filters are specified in polish notation where & is the Boolean and operator and | is the Boolean or operator. It is a good practice to restrict the searches to objectclasses of interest in order to minimize the duration it takes for a query to return. As a general rule of thumb: the more precise the query the shorter the response time.

Java-Browser

A Java LDAP browser with a simple GUI is available from http://www.globus.org/mds including extensive documentation and installation instructions.

CGI Browser

The GIS CGI Browser is a Perl script that allows you to browse the contents of the GIS through a normal web browser. The LDAP information is returned in html. The browser can be accessed from http://www.globus.org/mds.

LDAP in the Community

Since the GIS (MDS) is based on LDAP, it is possible to use a variety of application interfaces to communicate with the actual LDAP server. This software independence is one of the important features that make LDAP an ideal candidate for a very diverse compute environment as used in the Grid. The application interfaces available to access the GIS directly are for example:

Sh	shell scripts from the University of Michigan and Netscape
Perl	perlap, ldapperl, Net::LDAPapi
Java	Java APIs are available from Netscape and Sun http://java.sun.com/products/jndi
C	A C API can be found from Netscape and from OpenLdap/Univ. of Michigan

There are numerous books available that describe LDAP in much more detail. Please refer to them for more information.

Chapter 7 GASS: Remote Data

Summary of Chapter

GASS (Globus Access to Secondary Storage) provides programs and C APIs for remotely accessing data. This chapter describes how to start a GASS server, and how to use the globus-url-copy to transfer data to and from GASS servers. GASS programs allow data to be easily transferred from one machine to another.

globus-gass-server

The globus-gass-server program is used to make the data on one computer available to remote clients. To run, type:

```
% globus-gass-server
returns: https://evelyn.nas.nasa.gov:20143
```

This sets up your machine to be able to serve files to remote clients. The https URL that is output by the GASS server contains the host and port on which this server is listening for requests. It is used by clients to transfer data to and from this machine via the GASS server.

globus-url-copy

The globus-url-copy command is used to remotely access files from GASS servers, or from other servers that speak the http or https protocols (e.g. a web server). The general form of globus-url-copy is:

```
% globus-url-copy <fromURL> <toURL>
```

Examples:

```
% globus-url-copy https://evelyn.nas.nasa.gov:20143/tmp/foo file:/tmp/bar
% globus-url-copy https://evelyn.nas.nasa.gov:20143/tmp/foo -
```

In these examples, the *<fromURL>* is composed of the base URL that was returned by the globus-gass-server (https://evelyn.nas.nasa.gov:20143) followed by the path of file accessible to globus-gass-server (*/tmp/foo*). In the first example, the /tmp/foo file is remotely transferred from evelyn to a local file named /tmp/bar (file:/tmp/bar). In the second example, the single dash (-) pipes contents of the remote /tmp/foo file to stdout.

globus-rcp

The globus-rcp is the functional equivalent of the Unix rcp program, but uses GASS to perform its remote data transfer. You do not need to start a globus-gass-server by hand to use globus-rcp. Instead, globus-rcp will use GRAM to automatically start and stop remote GASS servers and clients for you. An example:

```
% globus-rcp evelyn.nas.nasa.gov:/tmp/foo /tmp/bar
```

Here the /tmp/foo file on evelyn is copied to the /tmp/bar on the machine running this command.

Chapter 8 RSL With Shell Script

Summary of Chapter

Shown here are an RSL file that will be submitted by Globus to pbs (Portable batch System), and a Bourne shell script, called by the RSL, that runs an application, OVERFLOW.

The RSL File

```
&   (count=4)
    (maxTime=20)
    (jobType=single)
    (directory=/scratch1/yarrow/X38.globus/)
    (executable=$(GLOBUS_TOOLS_PATH)/bin/globus-sh-exec)
    (arguments=run.sh)
    (stdout=run.out)
    (stderr=run.err)
    (environment=(GASS_URL $(GLOBUSRUN_GASS_URL)
        (OSNAME $(GLOBUS_HOST_OSNAME))
```

The executable here refers to a shell script that will be wrapped to be a PBS script.

The Shell Script

```
sourcefile=/scratch1/yarrow/X38-repository/all.tar
execprog=/ipg-home/yarrow/Overflow2.0.7/overflow2d.${OSNAME}

globus-url-copy ${GASS_URL}${sourcefile} - | tar xf -
globus-url-copy ${GASS_URL}${execprog} - | cat > overflow-exec
mpirun -np 4 overflow-exec
```

The shell script is getting mpirun to start Overflow2 (an overflow variant) as a 4 processor parallel job.

Chapter 9 Parallel Examples

Summary of Chapter

The goal of this chapter is to explain how to run an MPICH-G program using Globus. MPICH-G is not part of the standard Globus; it needs to be installed separately. See http://www.globus.org/mpi MPICH-G is an example of something at the "Application Toolkit" layer of the Integrated Grid Architecture (which we explained in Chapter 2). It is only one of many such application toolkits (albeit a useful one), and is included here to highlight the interesting capabilities that can be built on top of the underlying Grid Services.

If you would like to cut and paste the code in this chapter, see:
http://www.nas.nasa.gov/people/lou/globus/tryit-1.1.html

Preliminary Steps (For MPI examples)

Find out where your System Administrator has installed MPICH-G. MPICH-G is the Globus version of MPI (Message Passing Interface), a popular library for interprocess communication with parallel scientific programs. Specifically, you will need the compilation script mpicc —this is a script which will compile your MPICH-G C program. It eventually calls the system C compiler with the correct options for MPICH include files and globus libraries. MPICH-G is based upon MPICH, the popular implementation of the MPI standard developed by Gropp and Lusk at Argonne National Laboratory.

Scripts for FORTRAN and C++ are provided in the same directory as mpicc (mpif77, mpif90, etc.). This path will be treated as a shell environment variable called MPICH_BIN in these examples.

You must have obtained a proxy by executing grid-proxy-init.

Running an MPICH-G Program on the NAS IPG Testbed

This is an example of running an MPICH-G program in a specific environment.

A Basic MPI Example Using MPICH-G

The Message Passing Interface (MPI) is a popular library for interprocess communication with parallel scientific programs. MPICH-G is a supported MPI library for use with Globus. Below is a set of instructions for running an MPICH-G program on a single machine in the NAS IPG testbed.

Build the MPICH-G Program mpitest.c:

On evelyn.nas.nasa.gov, for example: a 32-bit (n32--see ABI(5) for more information) version of MPICH-G is installed in /ipg/mpich/mpich-g-n32.

To compile, execute the command:

```
/ipg/mpich/mpich-g-n32/bin/mpicc mpitest.c -o mpitest
```

[Note: To build a FORTRAN program, use either mpif77 or mpif90 in the same path as mpicc noted above.]

The mpirun Command

There are several ways to run an MPICH-G program with Globus. The preferred way is to use the mpirun command located in the /ipg/mpich/mpich-g-n32/bin directory:

First, you need to identify resources—machines where you want your job to run. This is specified in a file which is later read by mpirun. The file has the following format:

```
manager/resource-name [number-of-processes]
```

An example file called machines might look like:

```
evelyn.nas.nasa.gov/jobmanager-fork
```

The number-of-processes field is only necessary when running on multiple hosts. An example of this will be in the next section. mpirun uses the filename machines by default. If you would like to specify another filename, you need to use the -machinefile option to mpirun.

Finally, once you have compiled the program, and created the above machines file (in the same directory where you plan to execute the mpirun command), execute the command:

```
% /ipg/mpich/mpich-g-n32/bin/mpirun -np 4 mpitest
```

Your output should look similar to this:

```
Process #1 of 4 on host: evelyn at time: Mon Oct 11 11:49:48 1999
Process #0 of 4 on host: evelyn at time: Mon Oct 11 11:49:48 1999
Process #2 of 4 on host: evelyn at time: Mon Oct 11 11:49:48 1999
Process #3 of 4 on host: evelyn at time: Mon Oct 11 11:49:48 1999
```

Notes: What really happened here is that mpirun generated an RSL for you based on the number of processors, executable name, arguments, and machines file. It then ran globusrun on your behalf with that RSL. You could have generated your own RSL and submitted it directly using globusrun, but this is not preferred because the underlying syntax and/or mechanisms may change in the future.

The globus-job-run -: evelyn -np 4 'pwd'/mpitest command is similar to mpirun

Detailed information on mpirun and MPICH-G can be found in the MPI User's Guide: http://www-unix.mcs.anl.gov/mpi/mpich/docs/userguide/

An MPI Example on Two Hosts with Global Synchronization

The following example is intended to show how to run a parallel MPI job across two (extensible to more) machines. This example really starts to use the functionality of the Globus toolkit:

- Single sign-on to all hosts using grid-proxy-init with secure authentication.
- Global synchronization via something called DUROC. Basically, what this means is that within the MPI_Init() call, Globus will perform a global synchronization.

This example was run across Evelyn and Piglet. Note that Piglet is a Parallel Systems Group testbed machine and is not guaranteed to be reliable.

In order to run across multiple hosts, you simply specify all of them in a machines file. Ours looks like this:

```
evelyn.nas.nasa.gov/jobmanager-fork 4
piglet.nas.nasa.gov/jobmanager-fork 4
```

The second field (number-of-processes) can be used to tell mpirun how to distribute processes among machines. This file tells it to place the first four on evelyn, and second four on piglet.

After you have created the machines file, execute mpirun:

```
% /ipg/mpich/mpich-g-n32/bin/mpirun -np 8 mpitest
```

The output should look something like this:

```
Process #3 of 8 on host: evelyn at time: Mon Oct 11 12:58:43 1999
Process #2 of 8 on host: evelyn at time: Mon Oct 11 12:58:43 1999
Process #1 of 8 on host: evelyn at time: Mon Oct 11 12:58:43 1999
Process #0 of 8 on host: evelyn at time: Mon Oct 11 12:58:43 1999
Process #7 of 8 on host: piglet at time: Mon Oct 11 12:58:51 1999
Process #4 of 8 on host: piglet at time: Mon Oct 11 12:58:51 1999
Process #6 of 8 on host: piglet at time: Mon Oct 11 12:58:51 1999
Process #5 of 8 on host: piglet at time: Mon Oct 11 12:58:51 1999
```

The program (mpitest.c) is the same as in previous examples, so no re-compilation should be necessary.

Notes: One thing I have completely neglected to mention is how each host gets the executable (mpitest in this case). In the case of Evelyn and Piglet, home directories are shared (NFS mounted), so because I am running from my home directory, the executables are "automatically" there. If this were not the case, I would have to manually stage the executables to any host I wish to run on. If the hosts are of different architectures (e.g. SGI and Cray T3E), I must make sure I have the correct binaries staged as well. The next example will show how to use GASS to automatically stage executables.

Specifying additional attributes using an RSL

There are several reasons why the above instructions may not be adequate. For example, your resources may not share the same filesystem (or directory structure), or you may want to run through a job scheduler such as PBS and be able to specify scheduler attributes like memory and walltime.

In order to do these things, you need to specify your own RSL. mpirun can generate a template RSL for you, however. Here is an example:

1. Create a machines file with the following entries:

```
evelyn.nas.nasa.gov/jobmanager-pbs 4
piglet.nas.nasa.gov/jobmanager-fork 4
```

This example will run through the "PBS" job manager on Evelyn and the "fork" job manager on Piglet.

2. Execute mpirun with the -globusargs dumprsl option:

```
% /ipg/mpich/mpich-g-n32/bin/mpirun -np 8 -globusargs dumprsl mpitest >
mpitest.rsl
```

This should dump the RSL that mpirun would normally pass to globusrun and pipe it to a file. You can now edit the mpitest.rsl file adding appropriate attributes.

3. Edit the RSL (mpitest.rsl) adding a job attribute to the PBS part. Here is the updated RSL. (Note that the line beginning with the & has been wrapped to fit on this page; keep it all one line in your file.)

```
+
(
&(resourceManagerContact="evelyn.nas.nasa.gov:762/jobmanager-pbs:/C=U
S/O=Globus/O=NASA Ames Research Center/OU=Numerical
AerospaceSimulation/CN=evelyn.nas.nasa.gov-fork")
        (count=4)
        (maxtime=5)
        (label="subjob 0")
        (environment=(GLOBUS_DUROC_SUBJOB_INDEX 0))
        (directory=/u/lou/xxx)
        (executable=/u/lou/xxx/mpitest)
)
( &(resourceManagerContact="piglet.nas.nasa.gov:762/jobmanager-
fork:/C=US/O=Globus/O=NASA Ames Research Center/OU=Numerical
Aerospace Simulation/CN=piglet.nas.nasa.gov-fork")
        (count=4)
        (label="subjob 4")
        (environment=(GLOBUS_DUROC_SUBJOB_INDEX 1))
        (directory=/u/lou/xxx)
        (executable=/u/lou/xxx/mpitest)
 )
```

In the first segment of the above RSL the "maxtime" attribute has been added. For the PBS job scheduler on Evelyn, this sets the maximum amount of wallclock time the job will take (In this example, 5 minutes). Note: on some systems, maxtime refers to cpu time.

4. Now the new RSL is ready for submission. Use the following command:

```
% /ipg/mpich/mpich-g-n32/bin/mpirun -globusrsl mpitest.rsl
```

Note that you need not supply mpirun with the -np option because it is already reflected in the RSL file in the count attributes.

Your output should look similar to the following:

```
Process #7 of 8 on host: piglet at time: Tue Oct 12 16:47:06 1999
Process #6 of 8 on host: piglet at time: Tue Oct 12 16:47:06 1999
Process #4 of 8 on host: piglet at time: Tue Oct 12 16:47:06 1999
Process #5 of 8 on host: piglet at time: Tue Oct 12 16:47:06 1999
Process #1 of 8 on host: evelyn at time: Tue Oct 12 16:46:56 1999
Process #0 of 8 on host: evelyn at time: Tue Oct 12 16:46:56 1999
Process #2 of 8 on host: evelyn at time: Tue Oct 12 16:46:56 1999
Process #3 of 8 on host: evelyn at time: Tue Oct 12 16:46:56 1999
    [... PBS related job information]
```

Chapter 10 Globus Commands

Summary of Chapter

This chapter shows you the commands available in Globus 1.1. On your Unix command line, type a Globus command followed by the -usage option to get help. Note: in Globus 1.1, the first word of some of the commands has been changed from globus to grid. You will find information on the complete set of command-line programs in the Globus Toolkit at http://www.globus.org/v1.1/programs

See also the chapter on globus-job-run, globus-job-submit, mpirun, and globusrun in the chapter Running a Job.

Globus 1.1 Tools

The following tools are available in *<your-globus-tools-path>*in the Globus Toolkit ver 1.1. For usage information, including options available, type the command name with the **-usage** option. See also http://www.globus.org/v1.1/programs

Security

grid-cert-request	Creates a new certificate request and private key.
grid-cert-info	Displays certificate information.
grid-cert-renew	Creates a new key and renewal request for a Globus certificate.
grid-change-pass-phrase	Changes the "pass phrase" which protects the user's private key.
grid-proxy-init	Creates a proxy certificate, which can be used for authentication without having to enter the protecting pass-phrase.
grid-proxy-info	Displays proxy certificate information:
grid-proxy-destroy	Removes any user proxy certificates.

Job Submission

globusrun	Run a single executable on a remote site.
globus-setup-test	Verifies the user's setup of credentials.

globus-job-cancel	Cancels a job previously started using globus-job-submit.
globus-job-run	Allows the user to run a job at one or several remote resources. It translates the program arguments to a RSL request and uses globusrun to submit the job.
globus-job-clean	Kill the job if it is still running and clean the information concerning the job.
globus-job-get-output	For the job specified, get the standard output or standard error resulting of the job execution.
globus-job-status	Display the status of the job. See also globus-get-output to check the standard output or standard error of your job.
globus-job-submit	For batch job submission, that is, submitting a job to a queue via some local scheduling manager like PBS used NASA Ames.

Information Services

grid-info-add	Modifies the GIS server based on the contents of input file.
grid-info-remove	See grid-info-add.
grid-info-search	Searches the GIS.
grid-info-update	See grid-info-add.

Other Tools

globus-hostname	This is a simple shell script that acts like the Unix hostname.
globus-hostname2contacts	Converts a hostname to a list of resource manager contact strings.
globus-netstat	Hides the implementation-specifics of netstat and reformats the output to be consistent across architectures, producing a subset of UNIX System V netstat output.
globus-sh-exec	Sources the globus-sh-tools file, then executes a user script.
globus-version	Shows version number.
globus-development-path	Prints the full path to the "development" directory (include files and libraries) that corresponds best to the flavor indicated by the command-line options (pthreads, debug, 64-bit, support for SHM, ...)
globus-install-path	Prints the full path to the Globus install tree.
globus-rcp	Remote copy using GASS and Globus submission. Many options.

globus-tools-path	Prints the full path to the "tools" directory in the Globus installtree, tailored for the current architecture.
globus-services-path	Prints the full path to the "services" directory in the Globus install tree, tailored for the current architecture.
globus-tilde-expand	Expands the leading tilde sign ('~') (and the specified username user if provided) to the full path of the user's home directory.

Chapter 11 Bibliography and Reference

Web and Email Resources

- www.globus.org
  the central site
- support@globus.org
  to ask question related to installation of released versions of Globus and for application help
- bugs@globus.org
  to report bugs, etc.
- http://www.globus.org/documentation/faq.html
  Frequently Asked Questions. See also www.globus.org/documentation
- discuss@globus.org
  to ask question or discuss approaches in developing grid-aware applications via the globus tool kit. Subscribe by sending mail to majordomo@globus.org with a message = subscribe discuss
- developers@globus.org
  to ask questions related to development of future releases of Globus
- http://science.nas.nasa.gov/Software/p2d2/
  NAS debugging system for parallel and distributed programs

Technical Papers

Proc. 8th IEEE Symp. on High Performance Distributed Computing, "Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid", William E. Johnston, Dennis Gannon and Bill Nitzberg, 1999, HPDC8 [forthcoming!]

See http://www.globus.org/documentation/papers.html for a large selection of papers.

The Grid Book

The Grid: Blueprint for a New Computing Infrastructure, Edited by Ian Foster and Carl Kesselman: http://www.mkp.com/books_catalog/1-55860-475-8.asp

Chapter 12 Glossary

DUROC

Dynamically Updated Request Online Co-allocator

GASS

Global Access to Secondary Storage (remote file management)

GIS

Grid Information Service (formerly MDS, Metacomputing Directory Service) for locating and distributing characteristics of resources. Accesses the white pages and yellow pages served by LDAP. Used with a set of commands each beginning with grid-info

Globus

The Globus Project is a community effort, led by Argonne National Laboratory and the University of Southern California's Information Sciences Institute. Globus is developing the basic software infrastructure for computations that integrate geographically distributed computational and information resources

GRAM

Globus Resource Allocation Manager http://www.globus.org/gram

Grids

Widely distributed networks of high performance computers, stored data, instruments, and collaboration envir are widely distributed networks of high performance computers, stored data, instruments, and collaboration environments. See http://www.globus.org/testbeds

HBM

HeartBeat Monitor Is my process alive? The Globus Heartbeat Monitor (HBM) is designed to provide a simple, highly reliable mechanism for monitoring the state of processes. The HBM is designed to detect and report the failure of processes that have identified themselves to the HBM. See http://www-isi.globus.org/cgi/hbm-simple.cgi

Help

type any command with the -usage or -help option

LDAP

Lightweight Directory Access Protocol, for accessing information about hardware, software, and status in a networked environment. Directory-server software available from numerous sources.

Maxtime

Maximum time a job should be allowed to run. Refers to wallclock time on some systems, cpu time on others.

MDS

Former name of GIS.

MPI

Message Passing Interface, the industry-wide standard protocol for passing messages between parallel processors.

MPICH-G

Message Passing Interface for WANs; the Globus version of MPI

PBS

See http://pbs.mrj.com A flexible batch software processing; operates on networked, multi-platform UNIX environments, including heterogeneous clusters of workstations, supercomputers, and massively parallel systems.

Resources

Computers, instruments, and immersive environments; machines

RSL

Resource Specification List. A language that can be used with tools such as globusrun to specify resource requirements, such as what executable(s) to use, arguments to pass, etc.

usage

type any command with the -usage option for online help